Compilers

Project One

Write a complete lexer and parser that validates the source code and creates a symbol table for the following grammar:

```
Program       ::== Statement $
Statement     ::== P ( Expr )
              ::== Id = Expr
              ::== VarDecl
              ::== { StatementList }
StatementList ::== Statement StatementList
              ::== ε
Expr          ::== IntExpr
              ::== CharExpr
              ::== Id
IntExpr       ::== Digit Op Expr
              ::== Digit
CharExpr      ::== " CharList "
CharList      ::== Char CharList
              ::== ε
VarDecl       ::== Type Id
Type          ::== int | char
Id            ::== Char
Digit         ::== 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
Char          ::== a | b | c ... | z
Op            ::== + | -
```

Notes:

- The lexer is not as simple as our examples in class, so be careful.
- When you detect an error report it in helpful detail including where it was found.
- Include optional verbose output functionality that traces the stages of the parser including the construction of the symbol table. (An option for "Pirate" mode is okay, but better "Yoda" mode is. If you're considering GLaDOS mode, I am worried, but still alive. What about Run-DMC mode you ask? Well, it's tricky. HAL might make for an interesting compiler personality, if a little paranoid. Speaking of paranoid, error messages in the voice of Marvin the Paranoid Android would be, well, unique. Horror movie mode might be fun too.)

Create several test programs that cause as many different types of errors as you can in order to thoroughly test your code (think "code coverage"). Include several test cases that show it working as well. Write up your testing results in a document to hand in at the beginning of the class in which this is due. Be sure to note the names of all your team members.